From: <http://databases.about.com/od/specificproducts/a/keys.htm>

**Primary Keys/Foreign Keys**

As you may already know, [databases](http://databases.about.com/library/glossary/bldef-database.htm) use [tables](http://databases.about.com/library/glossary/bldef-table.htm) to organize information. (If you don’t have a basic familiarity with database concepts, read [What is a Database?](http://databases.about.com/od/specificproducts/a/whatisadatabase.htm)) Each table consists of a number of [rows](http://databases.about.com/library/glossary/bldef-row.htm), each of which corresponds to a single database [record](http://databases.about.com/library/glossary/bldef-record.htm). So, how do databases keep all of these records straight? It’s through the use of [keys](http://databases.about.com/library/glossary/bldef-key.htm).

**Primary Keys**

The first type of key we’ll discuss is the [primary key](http://databases.about.com/library/glossary/bldef-primarykey.htm). Every database table should have one or more [columns](http://databases.about.com/library/glossary/bldef-column.htm) designated as the primary key. The value this key holds should be unique for each record in the database. For example, assume we have a table called Employees that contains personnel information for every employee in our firm. We’d need to select an appropriate primary key that would uniquely identify each employee. Your first thought might be to use the employee’s name.   
  
This wouldn’t work out very well because it’s conceivable that you’d hire two employees with the same name. A better choice might be to use a unique employee ID number that you assign to each employee when they’re hired. Some organizations choose to use Social Security Numbers (or similar government identifiers) for this task because each employee already has one and they’re guaranteed to be unique. However, the use of Social Security Numbers for this purpose is highly controversial due to privacy concerns. (If you work for a government organization, the use of a Social Security Number may even be illegal under the Privacy Act of 1974.) For this reason, most organizations have shifted to the use of unique identifiers (employee ID, student ID, etc.) that don’t share these privacy concerns.   
  
Once you decide upon a primary key and set it up in the database, the database management system will enforce the uniqueness of the key. If you try to insert a record into a table with a primary key that duplicates an existing record, the insert will fail.   
  
Most databases are also capable of generating their own primary keys. [Microsoft Access](http://databases.about.com/library/glossary/bldef-access.htm), for example, may be configured to use the AutoNumber data type to assign a unique ID to each record in the table. While effective, this is a bad design practice because it leaves you with a meaningless value in each record in the table. Why not use that space to store something useful?

**Foreign Keys**

The other type of key that we’ll discuss in this course is the [foreign key](http://databases.about.com/library/glossary/bldef-foreignkey.htm). These keys are used to create relationships between tables. Natural relationships exist between tables in most database structures. Returning to our employees database, let’s imagine that we wanted to add a table containing departmental information to the database. This new table might be called Departments and would contain a large amount of information about the department as a whole. We’d also want to include information about the employees in the department, but it would be redundant to have the same information in two tables (Employees and Departments). Instead, we can create a relationship between the two tables.   
  
Let’s assume that the Departments table uses the Department Name column as the primary key. To create a relationship between the two tables, we add a new column to the Employees table called Department. We then fill in the name of the department to which each employee belongs. We also inform the [database management system](http://databases.about.com/od/administration/g/dbms.htm) that the Department column in the Employees table is a foreign key that references the Departments table. The database will then enforce [referential integrity](http://databases.about.com/cs/administration/g/refintegrity.htm) by ensuring that all of the values in the Departments column of the Employees table have corresponding entries in the Departments table.   
  
Note that there is no uniqueness constraint for a foreign key. We may (and most likely do!) have more than one employee belonging to a single department. Similarly, there’s no requirement that an entry in the Departments table have any corresponding entry in the Employees table. It is possible that we’d have a department with no employees.

**Definition:** Referential integrity is a database concept that ensures that relationships between tables remain consistent. When one table has a foreign key to another table, the concept of referential integrity states that you may not add a record to the table that contains the foreign key unless there is a corresponding record in the linked table. It also includes the techniques known as cascading update and cascading delete, which ensure that changes made to the linked table are reflected in the primary table.   
  
Consider the situation where we have two tables: Employees and Managers. The Employees table has a foreign key attribute entitled ManagedBy which points to the record for that employee’s manager in the Managers table. Referential integrity enforces the following three rules:

1. We may not add a record to the Employees table unless the ManagedBy attribute points to a valid record in the Managers table.
2. If the primary key for a record in the Managers table changes, all corresponding records in the Employees table must be modified using a cascading update.
3. If a record in the Managers table is deleted, all corresponding records in the Employees table must be deleted using a cascading delete.